Universitat de Lleida

TREBALL FINAL DE GRAU

Estudiant: Roger Bagué Masanés

Titulació: Grau en Enginyeria Informàtica

Títol de Treball Final de Grau: **Implementació i comparativa empírica de diferents metodologies de Processament de Llenguatge Natural (NLP) per al reconeixement d'entitats (NER) en textos jurídics**

Directors: Roberto García González i Mariano Garralda Barrio

Presentació

Mes: Juliol

Any: 2020

Resum

En aquest treball es pretén implementar i comparar empíricament diferents metodologies de buscar dates en textos d'àmbit jurídic.

Per fer-ho, s'han implementat diferents sistemes per tal d'extreure entitats de textos (en el nostre cas dates), comparar-los entre ells i a més a més, extreure'n conclusions.

Buscar entitats en textos a vegades pot resultar ambigu, per això a vegades s'han d'implementar sistemes basats en regles o sistemes estadístics. En aquest treball seran comparades les dues casuístiques.

Els sistemes basats en regles són:

- Sistema basat en expressions regulars.
- Sistema basat en regles de spaCy.

Aquests dos sistemes són capaços de detectar totes les dates que prèviament han estat observades i estudiades. En el nostre cas, troben totes les dates del text.

Els sistemes probabilístics són:

- Entrenament d'un model en blanc amb spaCy.
- Reentrenament d'un model existent amb *Transfer Leaning*, en el nostre cas tornarem a entrenar dos models.

Aquests sistemes tenen el següent rendiment:

RESULTATS DELS MODELS			
MODELS	MÈTRIQUES		
	Precision	*Recall*	*F1-Score*
MODEL EN BLANK	0.94	0.73	0.81
MODEL PETIT	**0.96**	**0.77**	**0.83**
MODEL MITJÀ	0.95	0.76	0.82

Taula 1 Resum dels resultats obtinguts dels models

Després de comparar tots els sistemes, la conclusió final és que el millor sistema per trobar dates en textos legals és el sistema basat en regles de spaCy, ja que ens dóna la possibilitat d'hibridar-lo amb les expressions regulars, cosa que ajuda en diversos casos molt específics i és un sistema capaç de trobar totes les dates.

Per altra banda, els sistemes probabilístics s'haurien d'utilitzar per trobar entitats les quals tinguin una mica d'ambigüitat. En el cas de les dates no n'hi ha.

Índex

Índex de taules

Índex de figures

Agraïments

Primerament m'agradaria donar les gràcies a la meva família per ajudar-me a arribar on estic ara, sobretot a la meva mare la qual m'ha ajudat a tirar sempre endavant i a continuar estudiant, ja que si no fos per ella crec que no hauria continuat amb els estudis.

També vull agrair al meu tutor de pràctiques, el Mariano Garralda per donar-me la idea de cap a on encarar el treball i per l'oportunitat que m'ha donat en formar part del seu equip a Indra. També al Roberto García, el qual m'ha ajudat molt en el desenvolupament del treball i ha estat sempre disponible per consultes tot i la situació actual degut a la Covid-19.

A tots els meus amics i amigues que m'han anat preguntant com he anat desenvolupant el treball.

Dono gràcies també a tot el professorat de l'EPS que he tingut, els quals m'han ensenyat tot el que sé i m'han donat els coneixements necessaris per arribar on sóc i a realitzar aquest treball, ser tan propers i tan professionals.

Finalment, a tots aquells professors de l'ESO i Batxillerat que algun cop em van menysprear, hem van tractar d'estúpid i van dir que mai arribaria a cap lloc. Gràcies ☺

1. Motivació

La principal motivació per dur a terme aquest treball final de grau ha estat el rol que he desenvolupat a l'empresa Indra, en la qual he estat assignat a un projecte de justícia que necessita coneixements de Processament de Llenguatge Natural per tal de dur-lo a terme.

Dins el projecte he tingut la tasca d'investigar nous *frameworks*, tècniques i llibreries per tal de saber com encarar el projecte en el futur i tenir diferents línies de visió.

Dins de les necessitats d'extracció d'entitats de característiques dins el projecte, necessitem extreure dates, quantitats i terminis.

La meva aportació a l'equip d'Indra han estat les conclusions d'aquest treball i la implementació dels sistemes que he fet, tots menys el basat en expressions regulars. Ara és decisió de l'equip el dia que es realitzarà la migració a aquests nous sistemes.

2. Objectius

L'objectiu principal d'aquest treball és implementar, comparar i raonar quina és la millor metodologia a l'hora de fer buscar entitats nomenades en textos jurídics, en el nostre cas, ens centrarem amb les dates.

Per buscar les entitats dins el text, s'utilitzaran diferents metodologies, les quals són:

1. Utilitzant expressions regulars.
2. Utilitzant un sistema basat en regles de la llibreria *spaCy*, mitjançant el *Part-Of-Speech tagging (POS),* entre altres.
3. Etiquetant un model en blanc i entrenant-lo, amb la llibreria *spaCy*.
4. Re entrenant un model ja existent, mitjançant *Transfer Learning*, amb la llibreria *spaCy*.

Un cop implementades aquestes diferents variants, es comprovà el rendiment i es farà una comparativa empírica dels resultats obtinguts, així com una valoració final amb les fortaleses i debilitats de cada implementació.

3. Estat de l'Art

3.1 Introducció

El Processament de Llenguatge Natural (NLP[1]) és un camp de la Intel·ligència Artificial (IA) el qual se centra a fer comprensible el llenguatge humà perquè les màquines el puguin entendre.

Aquesta branca es basa en una combinació de la informàtica i la lingüística per estudiar les regles i les estructures del llenguatge per crear sistemes intel·ligents capaços d'entendre, analitzar i extreure significat del text i la veu.

L'objectiu principal del NLP és construir un sistema computacional capaç de comprendre i/o generar llenguatge humà en totes les seves formes igual que ho faria una persona.

El NLP no tracta en la comunicació per mitjà de llengües d'una forma abstracta, sinó que es tracta en dissenyar mecanismes per comunicar-se que siguin eficaços computacionalment mitjançant programes que executin o simulen una comunicació. Els models aplicats es basen no només en la comprensió del llenguatge de per si, sinó també en aspectes generals cognitius humans.

Fins a la dècada de 1980, la majoria de sistemes de NLP es basaven en complexes regles estudiades i dissenyades a mà. A partir de finals del 1980, hi va haver una evolució molt gran gràcies a la introducció d'algoritmes d'aprenentatge automàtic (*Machine Learning*) per l'aprenentatge del llenguatge.

[1] Natural Language Processing

3.2 Història del Processament del Llenguatge Natural

El Processament de Llenguatge Natural va néixer a la dècada del 1960, com una branca de la intel·ligència artificial i la lingüística, amb l'objectiu d'estudiar els problemes derivats de la generació i comprensió automàtica del llenguatge natural.

La traducció automàtica, va ser la primera aproximació del NLP, on es van traduir més de 60 oracions del rus al anglès.

En els seus orígens, els mètodes que s'utilitzaven van tenir una gran acceptació i èxit, no obstant, quan aquestes aplicacions foren portades a la pràctica, en entorns no controlats i amb vocabularis genèrics, van començar a sortir moltes dificultats. Entres aquestes, hi ha els problemes de polisèmia i sinonímia, per la seva ambigüitat.

Cap als anys 80, la majoria de sistemes de NLP estaven basats en regles molt complexes escrites a mà. Va ser cap a finals dels 80 que es van introduir algoritmes de *Machine Learning* pel processament del llenguatge.

En els últims anys, les aportacions que s'han fet des d'aquest domini han millorat substancialment, permetent el processament de grans quantitats de d'informació en format text. Com a exemple podem observar: els motors de cerca en la web, les eines de traducció automàtica o la generació automàtica de resums.

3.3 Aplicacions del Processament del Llenguatge Natural

Els algoritmes de NLP solen basar-se en algoritmes d'aprenentatge automàtic. En lloc de codificar manualment enormes conjunts de regles, el NLP pot confiar en l'aprenentatge automàtic per aprendre aquestes regles automàticament analitzant un conjunt d'exemples i fent una inferència estadística. En general, com més dades analitzades, més precís serà el model. Aquests algoritmes poden ser utilitzats en algunes de les següents aplicacions:

Reconèixer entitats: Amb NLP podem identificar a les diferents entitats del text com ser una persona, lloc o organització. Aquest apartat serà més extensament explicat en l'apartat Reconeixement d'Entitats (NER)

Anàlisi de sentiment: També podem utilitzar NLP per identificar el sentiment d'una cadena de text, des de molt negatiu a neutral i a molt positiu.

Resumir textos: Podem utilitzar els models de NLP per extreure les idees més importants i centrals mentre ignorem la informació irrellevant, a més a més també es poden extreure tòpics per saber en què estan basats els textos.

Crear *chatbots*: Podem utilitzar les tècniques de NLP per a crear *chatbots* que puguin interactuar amb les persones.

Generar automàticament etiquetes de paraules clau: Amb NLP també podem fer una anàlisi de contingut aprofitant l'algoritme de LDA (*Latent Dirichlet allocation*) per assignar paraules claus a paràgrafs del text (*keywords*).

Classificació automàtica de textos en categories, es basa a detectar els temes recurrents i crear les categories.

3.4 Estratègies que es poden aplicar al NLP

El NLP es pot dividir en tres grans parts molts diferenciades, que serien: la part morfològica, sintàctica i semàntica.

Dins elles, es poden dur a terme diferents aplicacions o estratègies pel que fa al processament de textos.

3.4.1 Part morfològica

Consisteix en l'anàlisi intern de les paraules que formen oracions, es essencial per extreure la informació bàsica d'aquestes, com per exemple: la categoria sintàctica i significat lèxic

3.4.1.1 Truncament de paraules (Stemming)

Consisteix en tallar les paraules pel seu principi o final. Per fer-ho es té en compte una llista predefinida de prefixes i sufixes comuns que apareixen. La principal finalitat és reduir les paraules a la seva arrel.

Exemple:

L'arrel d'**amics -> amic**

3.4.1.2 Lematització de paraules (Lemmatization)

Consisteix en la segmentació d'una paraula per separar l'arrel (el lexema) dels morfemes de flexió.

Exemple:

El lemma de **cantàvem -> cantar**

La diferència principal que hi ha entre l'arrel que s'obté entre el procés d'*stemming* i *lemmatization* es:

L'*steming* utilitza mètodes heurístics que tallen els extrems de les paraules amb l'objectiu d'aconseguir l'arrel la majoria de les vegades i a vegades, inclou l'eliminació de prefixos.

El procés de *lemmatization* també intenta trobar l'arrel de la paraula però utilitzant l'ús del vocabulari i l'anàlisi morfològic de les paraules, amb la intencionalitat de retornar la forma bàsica de la paraula, coneguda com a *lemma*.

3.4.1.3 Segmentació morfològica (Morphological segmentation)

Consisteix a dividir les paraules en els morfemes, els quals són els elements més petits i significatius de les paraules.

Exemple:

El morfema de **tauleta -> taul-eta**

3.4.1.4 Etiquetatge gramatical (Part of speech tagging (POS))

Consisteix a l'etiquetatge gramatical de les paraules, de manera que a cada una d'aquestes se li assigna una categoria gramatical, com per exemple verb, substantiu, adjectiu, etc.

Exemple:

Va llençar l'ampolla ben lluny -> **Va llençar**(verb), **l'**(determinant), **ampolla**(nom), **ben**(adverbi), **lluny**(adverbi)

3.4.2 Part sintàctica

Consisteix en l'anàlisi de l'estructura de les oracions.

3.4.2.1 Toquenització (Tokenizer)

Es tracta de dividir (fer *split*) els textos en parts anomenades *tokens* els quals donen poden ser pel que fa a nivell de paraula (totes les paraules del text separades) o a frase, paràgraf, etc.

Serveix per poder treballar amb els segments de text que nosaltres considerem importants. S'ha de tenir en compte l' idioma, ja que cada idioma conté abreviatures i contraccions les quals poden provocar que el *tokenizer* ens faci un *split* on no volem.

Per exemple, en l'abreviatura 'Sr.'

3.4.2.2 Segmentació de paraules (Word segmentation)

Consisteix a resoldre el problema de dividir una cadena de text en paraules, molts cops un espai en blanc pot resultar una bona aproximació per aconseguir el nostre propòsit, però hi ha paraules les quals ens poden portar problemes, com per exemple les que tenen contraccions.

Exemple:

L'avió volava baix -> 'El', 'avió', 'volava', 'baix'

3.4.2.3 Anàlisi de dependències (Depencency Parsing)

Consisteix a reconèixer una frase, analitzar i assignar la seva estructura sintàctica. L'estructura més utilitzada és la d'arbre, la qual es pot generar utilitzant algoritmes d'anàlisi.

Exemple:

El guia mostrava els objectes als visitants->

El guia(subjecte), **mostrava els objectes als visitants** (predicat),

els objectes (complement directe), **als visitants** (complement indirecte)

3.4.3 Part semàntica

Consisteix en proporcionar la interpretació de les paraules una vegada eliminades les ambigüitats morfosintàctiques.

3.4.3.1 Desambiguació del sentit de les paraules (Word sense disambiguation)

Consisteix en assignar sentit a una paraula polisèmica (paraula la qual té més d'un significat) en funció del context en el qual es troba.

3.4.3.2 Generació de llenguatge natural (Natural Language Generation)

Consisteix a generar textos de forma automàtica que sigui llegible per l'esser humà, està molt relacionat a la generació d'informes per part de softwares empresarials.

3.5 Reconeixement d'Entitats (NER)

És un mètode d'extracció de noms d'entitats de textos que identifica automàticament aquestes i les classifica segons el tipus que són, els tipus més comuns són: persones, llocs i organitzacions.

L'extracció d'entitats és molt útil per analitzar text no estructurat. Permet obtenir informació clau per comprendre de què tracta un text.

Per desenvolupar un sistema capaç de reconèixer entitats, hi ha dues formes d'aconseguir-ho:

- **Sistemes basats en regles,** són deterministes i aquestes regles han estan estudiades i desenvolupades a mà, es necessita un estudi previ per tal de conèixer les diferents formes que pot tenir una entitat de ser nomenada i aquestes estan molt limitades, ja que una simple falta ortogràfica pot provocar que no es compleixi la regla. Aquests sistemes utilitzen Expressions Regulars, diccionaris i també altres sistemes híbrids que poden incorporar metodologies de NLP com el POS, *steemming*, *lemmatization*...

- **Sistemes basats en mètodes estadístics,** són estocàstics, estan basats en probabilitats de distribucions de les paraules, aquests utilitzen *Machine Learning*. Es basen en l'entrenament de models probabilístics pel reconeixement d'aquestes entitats.

Depenent de les entitats, depenent del mètode utilitzat es poden aconseguir millors *performances*, tot depèn de les entitats que volem extreure, ja que hi ha entitats molt definides les quals es poden treure fàcilment amb mètodes basats en regles, però hi ha altres entitats que es pot trobar ambigüitat i és millor un mètode probabilístic, ja que aquest es pot fixar en el context de l'entitat i determinar millor el resultat.

3.6 Metodologies d'avaluació del rendiment de models

Per tal d'avaluar els diferents sistemes d'extracció d'entitats, es tindran en compte els resultats obvinguts amb una matriu de confusió, on:

- **True Positive (TP):** element marcats com entitats i són entitats.
- **True Negative (TN):** elements no marcats com entitats i que no són entitats.
- **False Positive (FP):** elements marcats com a entitats i que no són entitats.
- **False Negative (FN):** elements no marcats com a entitats i que són entitats.

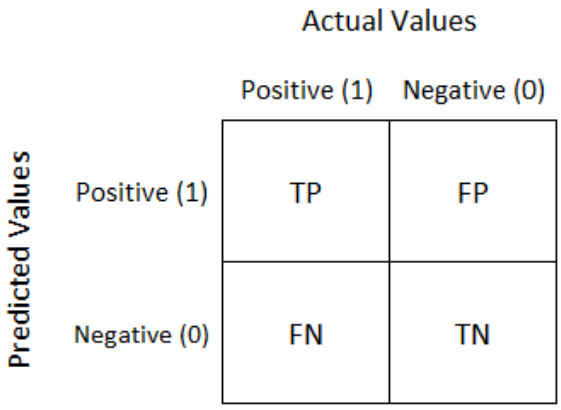

Figura 1: Matriu de confusió (Confusion Matrix)

I les mètriques utilitzades seran:

- ***Precision*:** és la relació entre les observacions positives que ha tingut el sistema i el total que aquest hauria d'haver trobat.

$$Precision = TP/TP+FP$$

- ***Recall*:** és la relació entre les observacions positives que ha tingut el sistema i totes les que ha tingut el nostre sistema.

$$Recall = TP/TP+FN$$

- ***F1-Score*:** és la relació general entre la *precision* i el *recall.* Té en compte tant els FP i el FN, això comporta una millor visió general del nostre sistema.

$$F1\text{-}Score = 2*(Recall * Precision) / (Recall + Precision)$$

3.7 Machine Leaning

Machine Learning és la ciència de programar sistemes perquè puguin aprendre ells mateixos de les dades.

3.7.1 Pipeline de *Machine Learning*

El flux d'un projecte de *Machine Learning* consta de les següents fases:

1. Recopilació de les dades.

   Es busca una o diverses fonts de dades per utilitzar en el nostre sistema.
2. Neteja de les dades.

   Es neteja, transforma i normalitza el conjunt de dades per tal que el siguin útils pel nostre sistema. Aquestes dades estan dividides en dos grans blocs:
   - El "*training set*" el qual consta aproximadament del 75% de les dades i s'utilitza per entrenar el model.
   - El "*validation set"* el qual consta d'aproximadament el 25% de les dades restants i s'utilitza per validar que el model entrenar funciona correctament.
3. Entrenament del model.

   S'entrena el model amb el "*training set".*
4. Validació del model.

   Es valida el model amb el "*validation set*".
5. Elecció del millor model.

   Un cop validats diversos models, es tria el que obté millors resultats.

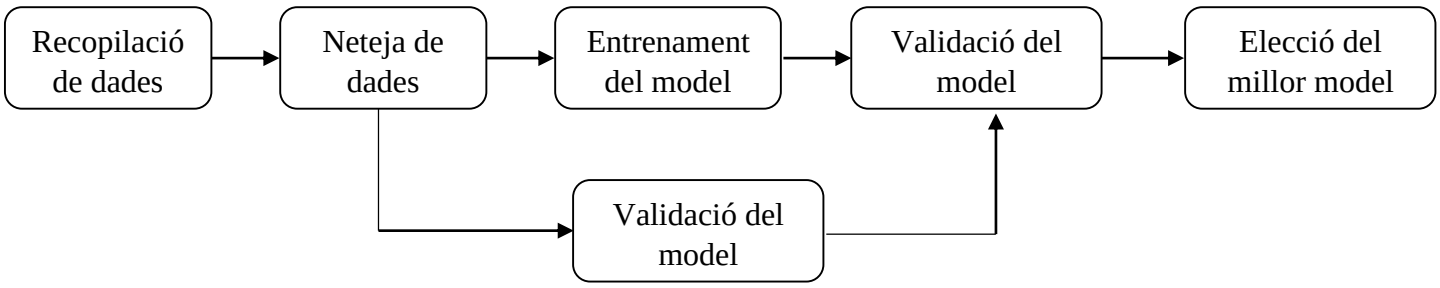

Figura 2: Flux general d'un projecte de Machine Learning

3.7.2 Tipus d'aprenentatge en Machine Learning

Hi ha tres categories d'aprenentatge en el *Machine Learning:*

1. Aprenentatge supervisat:

   S'utilitza un conjunt de dades que han estat prèviament etiquetades per tal d'entrenar el model.

2. Aprenentatge no supervisat:

   No s'utilitzen dades etiquetades, per tant, el sistema ha d'aprendre amb les característiques del "*training set*" com és un objecte.

3. Aprenentatge per reforç (*Reinforcement Learning*):

   El sistema aprèn per prova i error, es saben els resultats que volem obtenir des de bon principi, però no se sap com arribar a ells. L'algoritme va fent iteracions tenint en compte les eleccions escollides fins a arribar al resultat esperat.

3.7.3 Xarxes neuronals

Una xarxa neuronal és una estructura formada per un conjunt de nodes interconnectats que es comuniquen entre si, simulant la estructura i comportament de les neurones dels cervells humans.

Les neurones artificials, també anomenades **perceptrons** consten de tres components:

- Input (x_n):
  Rep informació procedent d'altres nodes amb els seus pesos determinats (w_n).
- Funció d'activació:
  Funció que determina si es genera o no un output.
- Output (Y):
  Informació que es transmet a altres nodes.

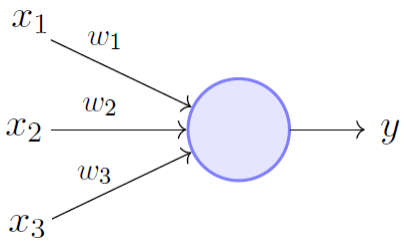

Figura 3: Perceptró

3.7.3.1 Estructura de les xarxes neuronals

Les xarxes neuronals consten de tres capes, les quals estan formades per un o més nodes:

- Capa dels inputs (input layer):
  És la capa encarregada de rebre els paràmetres d'entrada de la xarxa neuronal.
- Una o múltiples capes intermitges o ocultes (hidden layers):

Són una o un conjunt de capes les quals s'encarreguen de realitzar les operacions amb els valor de la capa de l'*input.*

- Capa dels outputs (output layer):
  Reben el resultat de les capes *hidden* i mostren el resultat obtingut

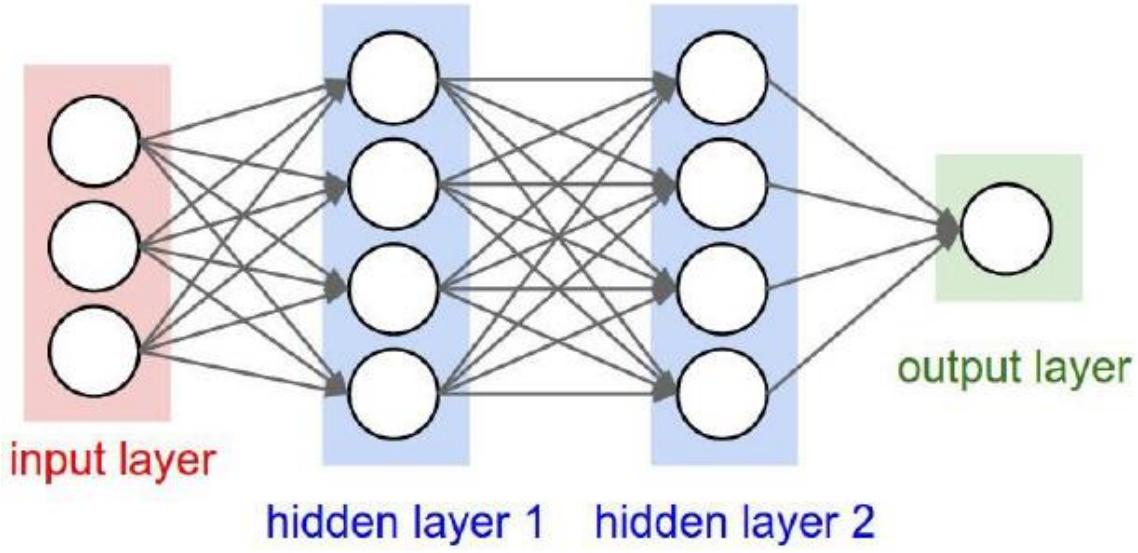

Figura 4: Estructura de les xarxes neuronals

3.7.3.2 Funcions d'activació

La funció d'activació és la que defineix si es produeix la sortida d'aquell node donada una entrada o conjunt d'entrades.

Hi ha diversos tipus de funcions d'activació, les més comuns són:

- Funció d'activació Binaria

  Pren les dades d'entrada i depenent si el valor d'aquestes està per sobre o per sota d'un valor determinat, aquesta s'activa o no, enviant la mateixa senyal a la següent capa

  - Inconvenients:
    No permet sortida per diversos valors

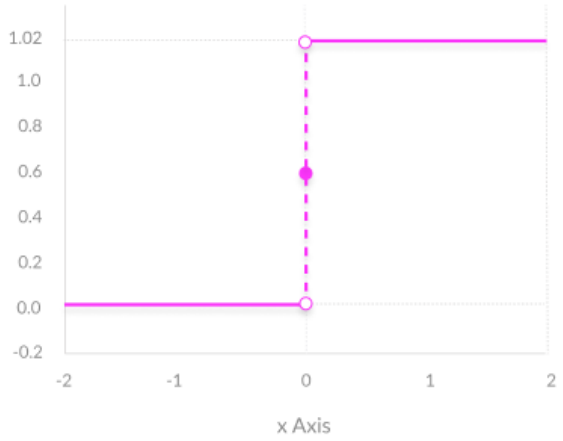

Figura 5: Funció d'activació binària

- Funció d’activació Lineal

  Pren les dades d’entrada, multiplicades pels pesos i crea una senyal de sortida.

  - Inconvenients:
    - No es possible utilitzar *backpropagation(gradient descent)* [2] per entrenar el model.
    - Totes les capes de la xarxa neuronal acaben en una sola, independentment de quantes capes *hidden* conté.

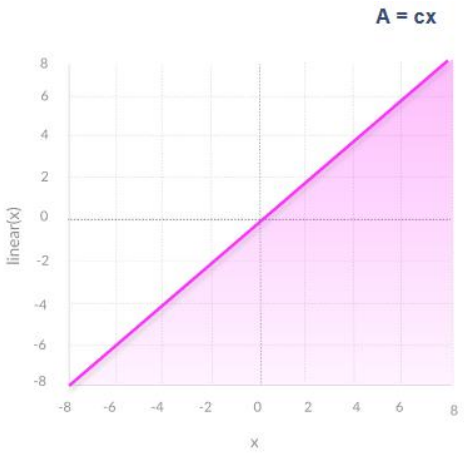

Figura 6: Funció d'activació lineal

- Funció d’activació Sigmoide
  - Avantatges:
    Gradient suau, evita salts en els valors de sortida, els quals son entre 0-1.
    Permet prediccions clares
  - Inconvenients:
    Gradient desaparegut, per valors alts o molt baixos, gairebé no hi ha canvi en la predicció, provocant que la xarxa no aprengui més o sigui massa lenta.

[2] Consisteix en recórrer la xarxa neuronal enrere per entendre quins pesos en els inputs poden donar una predicció millor.

Els *outputs* no es centren en el zero. Computacionalment car

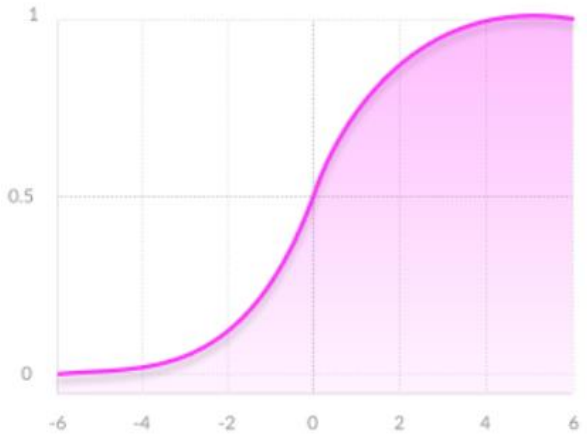

Figura 7: Funció d'activació sigmoide

- Funció d’activació TanH
    - Avantatges:
      Centrada en el zero, gradient suau, evita salts en els valors de sortida. Permet prediccions clares

    - Inconvenients:
      Gradient desaparegut, per valors alts o molt baixos, gairebé no hi ha canvi en la predicció, provocant que la xarxa no aprengui més o sigui massa lenta.
      Els *outputs* no es centren en el zero. Computacionalment car

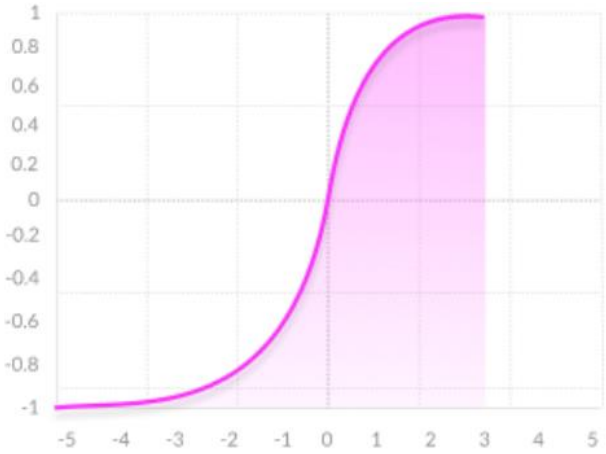

Figura 8: Funció d'activació TanH

- Funció d’activació ReLU
    - Avantatges:
      Computacionalment eficient, permet a la xarxa convergir ràpidament

No lineal, té una funció derivativa la qual permet *backpropagation*.

- Inconvenients:
  Quan les entrades s'aproximen a zero o són negatives, el gradcient de la funció es converteix de zero, la xarxa no pot realitzar *backpropagation* i aprendre.

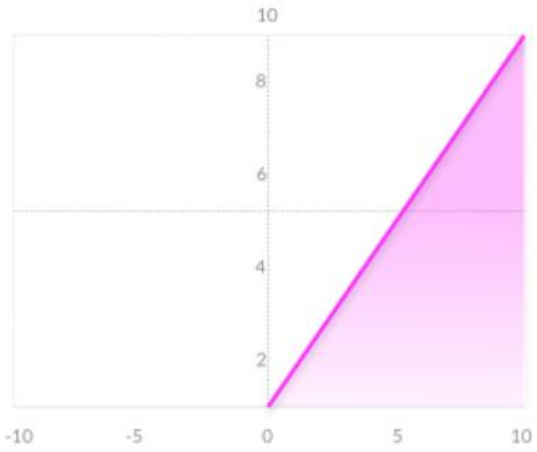

Figura 9: Funció d'activació ReLU

3.7.3.3 Xarxes neuronals utilitzades i tecnologies associades a l'entrenament del model d'spaCy

El sistema NER de spaCy utilitza una xarxa neuronal convolucional profunda amb connexions residuals (deep convolution neural network(CNN) with residual connections) amb encapsulament "Bloom"(Bloom embedding).

L'autor de spaCy Matthew Honnibal, ha decidit utilitzar una CNN (tot i que la majoria de cops aquesta s'utilitza per al tractament d'imatges), ja que els resultats obtinguts són més que satisfactoris.

L'algoritme aplica un filtre convolucional sobre el text d'entrada per predir com les paraules següents poden canviar les etiquetes de les entitats actuals, cal dir, que aquesta filtre convolucional també s'utilitza pel POS, per l'anàlisi de dependències a més del NER.

Les paraules que vindran poden canviar l'entitat, reduir-la (fer-la més granular) o produir l'entitat
Les paraules d'entrada son encapsulades mitjançant "Bloom embedding", el qual crea un vector amb la normalització, prefix, sufix i el POS de cada paraula.

Les connexions residuals són usades, ja que això la sortida de cada capa convolucional és la suma d'aquesta sortida i l'entrada.

Aquest sistema està dissenyat per aportar un bon balanç d'eficiència, precisió i adaptabilitat.

3.7.4 Transfer learning

És el procés pel qual és re entrena un model que ja fa una determinada tasca per fer-ne una altra de similar.

Transfer learning és una drecera per estalviar temps i obtenir un rendiment millor, però el benefici d'utilitzar-lo no se sap fins que el model no ha estat re entrenat i avaluat.

A vegades no hi ha moltes dades per entrenar un model propi, és llavors quan el transfer leaning et pot ajudar a desenvolupar models de qualitat.

3.8 Eines de NLP

Seguidament, s'expliquen les principals característiques de llibreries i *frameworks* que s'utilitzen per al NLP, a més a més es farà una petita comparativa entre elles i en quins casos és millor utilitzar una o una altra.

3.8.1 Natural Language Toolkit (NLTK)[3]

És una plataforma que permet crear programes en Python per al processament de llenguatge natural. Proporciona interfícies fàcils d'utilitzar i a més, 50 corpus i lèxics, com *WordNet*, juntament amb un conjunt de biblioteques de processament de text per a la classificació, la toquenització, l'*stemming*, l'etiquetatge, l'anàlisi i el raonament semàntic i un fòrum de debat molt actiu.

3.8.2 Stanford CoreNLP[4]

És una eina que proporciona un conjunt d'eines per al processament del llenguatge natural. Pot donar les formes simples de paraules, les seves formes sintàctiques, també permet dir si són noms d'empreses, persones, etc., normalitzar dates, hores i quantitats numèriques, marcar l'estructura de les oracions en les frases i les seves dependències sintàctiques. També permet dir quines frases fan referència a entitats, anàlisi de sentiment de frases, extreure relacions, obtenir citacions que la gent diu, etc.

3.8.3 spaCy[5]

spaCy és una llibreria de software lliure, gratuïta pel Processament de llenguatge natural (NLP) en Python. Està dissenyat específicament per l'ús en producció i ajuda a crear aplicacions que processen i 'entenen' grans volums de text. Pot ser utilitzat per extreure informació o per fer sistemes que entenguin el llenguatge natural, també serveix per preprocessar text per *deep-learning*.

[3] Pàgina web: https://www.nltk.org/
[4] Pàgina web: https://nlp.stanford.edu/
[5] Pàgina web: https://spacy.io/

3.8.4 Allen-NLP[6]

És una llibreria de Processament de Llenguatge Natural d'investigació Apache 2.0, construïda sobre PyTorch, per desenvolupar models *state-of-the-art* d'aprenentatge profund en una gran varietat de tasques lingüístiques.

AllenNLP està construït i mantingut per l'Allen Institute per la Intel·ligència Artificial, en estreta col·laboració amb investigadors de la Universitat de Washington, entre altres llocs. Amb un bon equip d'investigadors i enginyers de programari, el projecte AllenNLP està situat en una posició única per al creixement a llarg termini al costat d'una comunitat de desenvolupament de codi obert.

3.8.5 Comparació de característiques entre les eines

	SPACY	**NLTK**	**CORENLP**	**ALLEN-NLP**
Llenguatge de programació	Python	Python	Java/ Python	Python
Models amb xarxes neuronals	SI	NO	SI	SI
Vectors de paraules integrats	SI	NO	NO	SI
Suport a més d'un idioma	SI	SI	SI	SI
Toquenització[7]	SI	SI	SI	SI
***Part-of-speech Tagging*[8]**	SI	SI	SI	SI
Segmentació d'oracions[9]	SI	SI	SI	SI
Anàlisis de dependències[10]	SI	NO	SI	SI
Reconeixement d'entitats[11]	SI	SI	SI	SI
Relació d'entitats[12]	SI	NO	NO	NO

Taula 2 Comparació de característiques entre les eines

[6] Pàgina web: https://allennlp.org/
[7] Veure secció 3.4.2.1
[8] Veure secció 3.4.1.4
[9] Veure secció 3.4.2.2
[10] Veure secció 3.4.2.3
[11] Veure secció 3.5
[12] Característica que relaciona una mateixa entitat NER a un identificador únic.

3.8.6 Quan s'hauria d'utilitzar cada eina?

	SPACY	NLTK	CORENLP	ALLEN-NLP
Sóc un principiant que just comença en el NLP	SI	SI	SI	NO
Vull construir aplicacions en producció *end-to-end*	SI	NO	NO	NO
Vull provar diferents arquitectures de xarxes neuronals pel NLP	NO	NO	NO	SI
Vull provar els models més nous amb molta *accuracy*	NO	NO	SI	SI
Vull entrenar els meus models amb les meves dades pròpies	SI	SI	SI	SI
Vull que la meva aplicació sigui eficient en CPU	SI	SI	NO	NO

Taula 3 Casos d'ús de cada eina

4. Desenvolupament del projecte

El projecte s'ha fet amb diverses iteracions, en cada una d'elles s'han estudiat i implementat els diferents sistemes que en aquest projecte s'expliquen.

4.1 Requeriments generals

Els sistemes han de:

- Ser capaços de preprocessar textos legals.
- Ser capaços de trobar totes les dates dins el text.
- Ser capaços de donar un resultat final després de la seva execució.
- Ser capaços de passar els testos implementats.

4.2 Requeriments no funcionals

Els sistemes han de:

- Ser compatibles amb diversos entorns.
- Fàcilment mantenibles.
- Tenir un bon rendiment.

Pel que fa a la implementació del projecte, el llenguatge de programació escollit ha estat: **Python 3.8**

Python ha estat escollit per què la llibreria spaCy l'utilitza i, a més a més, és un llenguatge amb el qual és fàcil i ràpid de programar.

L'entorn de desenvolupament escollir ha estat **PyCharm,** ja que ja havia treballat abans amb ell i resulta molt còmode i té una integració amb el llenguatge de programació Python molt bona.

4.3 Organització del projecte

Per ficar en context el desenvolupament del projecte, primer he d'explicar com he organitzat el projecte, el qual està publicat a Github[13]:

Dins el projecte hi podem trobar diferents carpetes: documents, labeler_and_converter_of_data, models i test.

[13] URL del projecte Github: https://github.com/rbague5/tfg-nlp

En la carpeta de documents hi ha tots els textos amb els quals s'han fet les proves i entrenat els models.

En la carpeta labeler_and_converter_of_data hi trobem un etiquetador automàtic de textos i també un checker, per comprovar que els etiqueti bé.

En la carpeta models, podem trobar les diferents metodologies implementades per tal de buscar entitats, dins hi ha les carpetes custom_model i spacy. Dins cada una d'elles hi trobem les diferents implementacions per abordar el problema principal: Decidir quin és el millor mètode per aplicar l'algoritme NER a textos jurídics.

Finalment, en la carpeta de test, podem veure els testos per comprovar el rendiment que tenen les diferents implementacions i també observar els resultats obtinguts.

5. Metodologia

Per cada tipus d'implementació, s'ha hagut de fer un estudi previ per veure si la solució seria apropiada pel tipus de projecte que s'ha realitzat, després petites proves unitàries i finalment una implementació final utilitzant la tecnologia en qüestió que es volia testejar.

Les diferents metodologies tenen implementada una estructura en comú, el qual seria un *pipeline* típic d'un projecte de *Machine Learning*:

5.1 Recol·lecció de dades:

Els textos judicials que he utilitzat pel desenvolupament del projecte han estat descarregats del CENDOJ, Centro de Documentación Judicial[14]

5.2 Neteja i preparació de les dades:

Com els documents descarregats han estat en format .pdf, utilitzant una eina *online* de OCR[15] aquests han estat passats a format .txt

Un cop estan en el format desitjat, el següent pas de neteja ha estat eliminar els salts de línia, els títols dels documents i els números de pàgina. A més a més, com l'eina OCR no és del to precisa i també els documents contenen dobles espais en blanc i fins i tot més, aquestes també han estat substituïts per només un espai.

5.3 Implementació core del algoritme:

Cada implementació té una funció amb l'algoritme en si que busca les entitats dins el text, en aquesta funció rep el text preprocessat com a paràmetre i retorna els resultats que ha trobat.

5.4 Post procés (si escau):

Hi ha vegades que es necessita un postprocés per retornar els resultats correctament, per exemple en el cas del NLP podria ser: marcar els indexes d'inici i fi de les entitats que es troben en el text o també la neteja d'aquestes entitats en si, com signes de puntuació, etc.

[14] Pàgina web: http://www.poderjudicial.es/search/indexAN.jsp
[15] Optical Character Recognition

6. Planificació Temporal

El projecte va ser iniciat el 16 de gener, quan es va decidir quina seria la temàtica ha desenvolupar. Però inicialment hi va haver un estudi previ del món del processament de llenguatge natural, amb un estudi de les eines que hi havia i les diferents llibreries.

Aquesta feina prèvia és la que vaig realitzar a l'empresa Indra i que va començar a mitjans del mes d'octubre.

El final del projecte s'estima que serà aproximadament pels 23 de juny, dia el qual acaben les meves pràctiques extracurriculars a l'empresa.

7. Pressupost

El projecte s'ha realitzat amb un ordinador de sobretaula, dues pantalles i 550 hores de treball.

El cost de les hores s'ha tingut en compte pel conveni que hi ha de pràctiques extracurriculars.

Les llibreries utilitzades i IDE són gratuïts.

MATERIAL	COST
Ordinador	1000 €
2x Pantalles AOC 24"	300 €
550h x 10€/h	5500 €
TOTAL	**6800 €**

Taula 4 Pressupost del projecte

8. Sistema basat en expressions regulars

Aquest sistema utilitza expressions regulars, per tant, és un sistema basat en regles les quals han estat estudiades, dissenyades i escrites basades en l'observació i estudi dels diferents patrons que s'han observat en els diversos textos legals que s'han llegit.

Cal esmentar que aquest sistema no es troba en el projecte, ja que forma part del codi del projecte en l'empresa Indra en el qual estic treballant i no puc compartir el codi.

És per això, que només mostraré els resultats i les conclusions d'aquest sistema, en l'apartat 11.1 Expressions Regulars.

9. Sistema basat en regles de spaCy

Aquest sistema està basat en regles les quals han estat estudiades, dissenyades i escrites basades en l'observació i estudi dels diferents patrons que s'han observat en els diversos textos legals que s'han llegit. Està basat en la part del reconeixement d'entitats mitjançant el *Part-Of-Speech tagging* de la llibreria spaCy.

9.1 Especificació

El sistema ha de ser capaç de trobar totes les dates d'un text legal mitjançant les regles que han estat estudiades i implementades.

Aquestes regles estan fetes amb un motor de cerca de la llibreria spaCy el qual ens dóna unes característiques i un nivell de sofisticació i precisió que no ens donaven les expressions regulars.

9.2 Disseny

El *pipeline* de tots els sistemes té la mateixa arquitectura general, però cada sistema té unes petites variacions que el fan únic. En el cas d'aquest sistema, el disseny és el següent:

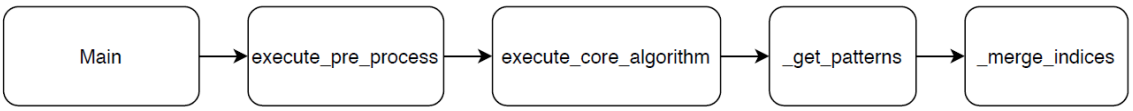

Figura 10 Pipeline del sistema basat en regles de spaCy

9.3 Implementació

La funció **main** s'encarrega de llegir el fitxer, passar-lo a la funció **execute_pre_process** i un cop el fitxer ha estat preprocessat, aquest es passa a la funció **execute_core_algorithm** i finalment es mostren els resultats obtinguts del fitxer que hem llegit.

En la funció **execute_pre_process** el primer que s'ha fet ha estat separar tot el text en una llista de frases d'aquest, de manera que el text serà netejat frase a frase. Tot i que aquest al final serà unit tot en un text per què el **core_algoritme** és capaç d'executar-lo tot a la vegada.

Per fer una segmentació s'ha utilitzat el paquet PunktSentenceTokenizer de NLTK, ja que era el que m'oferia el que realment estava buscant. El problema principal que hi ha en la segmentació de frases, és que els algoritmes acostumen a fer la segmentació tenint en compte els punts que hi ha en les frases, llavors hi ha moltes abreviatures que poden causar que les frases no estiguin realment ben separades.

En el cas dels textos jurídics s'utilitzen moltes abreviatures i aquestes són les excepcions que s'han vist i afegit en una llista per tal que el sistema segmenti bé el text:

```
abbreviation =
[
'sra', 'dª', 'dña', 'sras', 'sres', 'sr', 'excmos', 'excmo', 'excma',
'excmas', 'ilma', 'ilmas','ilmo', 'ilmos', 'ilma', 'ilmas', 'art',
'arts', 'núm', 'cp', 'c.p', 's.l', 'rcud', 'rcuds','rec'
]
```

Un cop tenim tot el text segmentat, s'eliminen elements del text que no ens aporten res, com per exemple: múltiples espais en blanc, símbols estranys, títols del document i números de pàgina.

Finalment, com s'ha mencionat abans, el text s'ajunta tot un altre cop.

Cal mencionar, que el fet de separar el text per frases pot ser utilitzar en un futur en cas que s'utilitzi un volum molt gran de dades, ja que això ens ajuda a mantenir uns indexes no molt elevats a l'hora d'identificar entitats i a més a més, també ens pot ajudar si en un futur es vol paral·lelitzar el sistema.

En la funció **execute_core_algorithm** rep com a paràmetre el text netejar i s'encarrega d'executar l'algoritme implementat, finalment retorna el resultat en format de llista on hi ha totes les dates trobades en el text. Dins aquesta funció, es carrega el model que volem utilitzar pel que fa al POS del text i després es crea una instancia del objecte Matcher en el qual s'afegeixen els patrons que s'han estudiar i creat del que nosaltres volem trobar en el text. Aquest es carregent amb la funció **_get_patterns** .

La especificació dels atributs que hi ha en el model que carreguem i que assigna spaCy a les paraules i podem utilitzar per escriure els patrons és:

ATRIBUT	TIPUS	DESCRIPCIÓ
ORTH	unicode	El text exacte d'un *token*
TEXT	unicode	El text exacte d'un *token*
LOWER	unicode	El text en minúscula d'un *token*
LENGTH	int	La llargada del *token*
IS_ALPHA, IS_ASCII,IS_DIGIT	bool	El *token* consisteix en caràcters alfanumèrics, ASCII o dígits.
IS_LOWER, IS_UPPER, IS_TITLE	bool	El *token* és majúscula, minúscula o títol
IS_PUNCT, IS_SPACE	bool	El *token* és un signe de puntuació o espai
LIKE_NUM, LIKE_URL, LIKE_EMAIL	bool	El *token* és com un numero, URL o e-mail
POS, TAG, DEP, LEMMA, SHAPE	unicode	La forma del *token* del POS, TAG, DEP, LEMMA i SHAPE
ENT_TYPE	unicode	El *token* és una entitat
_	dict	El *token* és una propietat personalitzada

Taula 5 Atributs disponibles pels tokens

A més a més dels atributs disponibles pels *tokens*, també hi ha atributs per aquests atributs i patrons que es poden aplicar a aquests, que son:

ATRIBUT	TIPUS	DESCRIPCIÓ
IN	qualsevol	L'atribut forma part d'una llista
NOT_IN	qualsevol	L'atribut no forma part d'una llista
==, >=, <=, >, <	qualsevol	El valor de l'atribut es igual, més gran o igual, més petit o igual, més gran o més petit
REGEX	Expressió regular	Serveix per buscar l'expresió regular d'un atribut d'un token determinat

Taula 6 Patrons de sintaxi i atributs pels tokens

Uns exemples dels patrons que s'han utilitzat són:

```
pattern1 = [{'POS': 'NUM', 'IS_DIGIT': True}, {'POS': 'ADP'}, {'ORTH': {'IN': meses}},{'POS': 'ADP'},{'POS': 'NUM', 'IS_DIGIT': True, 'LENGTH': {"==": 4}}]
```

El qual serviria per trobar, per exemple: 12 de mayo de 2015

```
pattern2 =  [{'POS': 'NUM', 'IS_DIGIT': True}, {'POS': 'ADP', 'LEMMA': 'de'}, {'ORTH': {'IN': meses}}
```

Serviria per trobar, per exemple: 12 de mayo

Com es pot observar s'utilitza l'ORTH, el qual serveix per trobar un text exacte i li passa una llista declarada inicialment amb tots els mesos de l'any, **això fa que sigui pràcticament impossible que el patró no sigui una data.**

En una primera aproximació en comptes de passar una llista, es buscava que el POS fos un NOUN, ja que els mesos els marca així, però hi havia cops que trobava coses que no eres dates.

Finalment, la funció _merge_indices comprova que no hi hagi elements que se superposen, i en cas que n'hi hagi agafa el més gran, això passa degut a què hi ha patrons que tenen la mateixa forma, per exemple:

12 de enero de 2019

12 de enero

Tenim dos patrons que troben aquest text, degut a què són formats de dates molt comunes, però aquesta data és només una i volem la llarga.

Els resultats i les conclusions d'aquest sistema, es poden veure en l'apartat 11.2 spaCy

10. Entrenament d'un model amb spaCy

10.1 Especificació

S'ha d'entrenar un model de zero el qual sigui capaç de trobar dates en un text, per fer-ho primer s'ha d'etiquetar un text amb les entitats que volem trobar i després passar aquest *training set* al codi de spaCy per tal d'entrenar la xarxa neuronal.

10.2 Disseny

El disseny del sistema d'entrenar el model és el mateix que es mostra en aquest link, el qual és el següent:

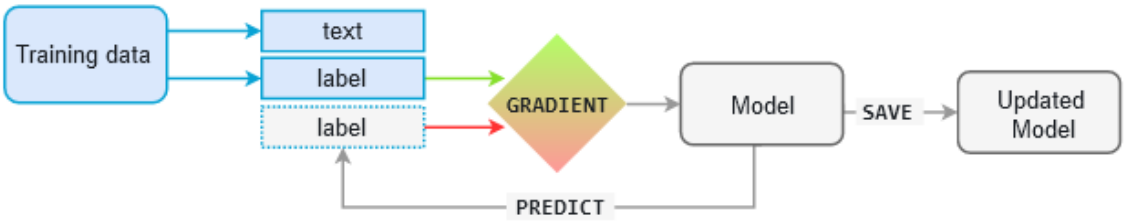

Figura 11 Pipeline del entrenament d'un model amb spaCy

10.3 Implementació

Els passos a dur a terme a l'hora d'entrenar un model en spaCy són els següents:

1. S'ha de **carregar el model** amb el qual es vol començar, o d'altra banda s'ha de crear un **model en blanc** utilitzant `spacy.blank` amb l'ID del llenguatge que volem utilitzar. Si utilitzem un model en blanc, s'ha d'afegir l'algoritme NER al *pipeline*. En cas que estiguem utilitzant un model existent, s'han de deshabilitar tots els altres components del *pipeline* mentre l'estiguem entrenant, utilitzant `nlp.disable_pipes`. D'aquesta manera només entrenarem el reconeixement d'entitats (NER).

2. **Afegir la nova entitat** al NER utilitzant el mètode `add_label`. Es pot accedir al NER en el *pipeline* mitjançant `nlp.get_pipe('ner')`.

3. **Iterar sobre els exemples etiquetats** i cridar el mètode `nlp.update` que llegeix les paraules d'entrada. Per cada paula fa una **predicció.** Seguidament, fa una consulta a les entitats etiquetades per si ho ha fet correctament. En cas que ho

hagi predit incorrectament, ajusta els pesos de la xarxa neuronal perquè així en la següent iteració aconsegueixi una millor aproximació.

4. **Guarda** el model entrenat utilitzant el mètode `nlp.to_disk`.

5. **Testeja** el model per comprovar que les entitats trobades han estat reconegudes correctament.

10.4 Entrenament d'un model en blanc amb spaCy

10.4.1 Especificació

Per entrenar un model, primerament s'han d'etiquetar textos amb les entitats que volem trobar, en el meu cas he realitzat un etiquetador automàtic el qual gràcies al sistema basat en regles de spaCy busca entitats frase a frase i les etiqueta de forma automàtica. El resultat el guarda en format .json amb la següent estructura:

```
[
{"content": "El expediente de contratación fue incoado el 24 de julio de 2017 por el representante de la Delegación, por un valor estimado de 167.065 euros (IVA incluido), y por una duración del contrato de 3 meses.","entities": [[45, 64, "FECHA"]]},
{"content": "Constan como pagados los dos primeros, no así el último; y se adjuntan como justificantes, las copias del resguardo de las dos transferencias bancarias y su contabilización en el programa GAE (Gestión de Autonomía Económica).", "entities": []}
]
```

Com es pot observar, el "content" conté la frase etiquetada i l'"entities" conté una llista amb totes les entitats que ha trobat, això com l'índex d'inici i fi d'aquesta entitat i el tipus d'entitat que és. A més a més en les frases que no hi ha entitats també s'han de ficar, perquè aprengui a vegades pot ser que no n'hi hagi.

10.4.2 Disseny

El disseny del sistema general del sistema funciona igual que el sistema basat en regles de spaCy exepte el **execute_core_algorithm:**

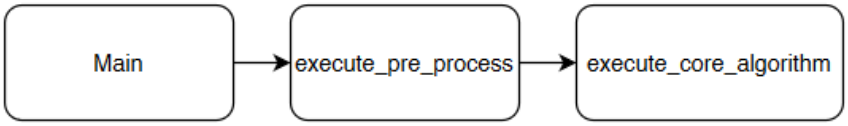

Figura 12 Pipeline del sistema per provar un model entrenat

10.4.3 Implementació

En el **execute_core_algorithm** el que es fa és carregar el model que volem provar, en el nostre cas el model **blank_model** i iterem pel `document.ents` per veure totes les entitats que ha trobat el model en el nostre text en particular. Per filtrar-les en el nostre cas son les `ent.label_ == 'FECHA'`

Amb aquest codi podem veure les dates trobades en el text:

```
for ent in document.ents:
    if ent.label_ == 'FECHA':
        print(ent.text)
```

10.4.4 Resultats

La següent taula mostra els resultats del model en blanc provats amb els diversos documents del *validation set.*

RESULTATS MODEL EN BLANC			
TEXTOS	METRIQUES		
	Precision	*Recall*	*F1-Score*
ej_0_es	1.00	0.86	0.93
ej_1_es	0.91	0.38	0.54
ej_2_es	1.00	0.94	0.97
ej_3_es	0.85	0.74	0.79
Mitjana dels textos	**0.94**	**0.73**	**0.81**

Taula 7 Resultats del rendiment del model en blanc

10.5 Reentrenament d'un model existent amb *Transfer Leaning*

10.5.1 Especificació

Per entrenar un model, primerament s'han d'etiquetar textos amb les entitats que volem trobar, en el meu cas he realitzat un etiquetador automàtic el qual gràcies al sistema basat en regles de spaCy busca entitats frase a frase i les etiqueta de forma automàtica. A més a més de les entitats que volem afegir noves al model, s'han d'etiquetar també les entitats que abans el model ja era capaç de trobar, ja que si no ho fem, pot passar un problema conegut com a **Catastrophic forgetting**. El que passa és que si només entrenem el model amb les noves entitats, pot ser que el model oblidi les entitats que ja coneixia, és per això que hem d'etiquetar també les entitats que ja es trobaven.

10.5.2 Disseny

El disseny del sistema general del sistema funciona igual que el sistema basat en regles de spaCy exepte el **execute_core_algorithm:**

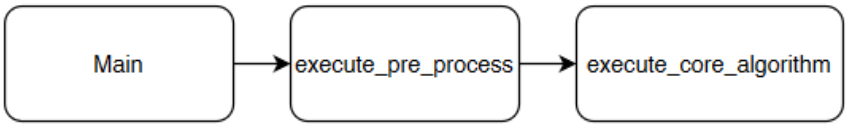

Figura 13 Pipeline del sistema per provar un model entrenat

10.5.3 Implementació

En el **execute_core_algorithm** el que es fa és carregar el model que volem provar, en el nostre cas els models són **es_core_legal_sm** o **es_core_legal_md**, aquests models son creats a partir dels models **es_core_news_sm** i **es_core_news_md** respectivament.

Un cop carregats en el sistema, iterem pel `document.ents` per veure totes les entitats que ha trobat el model en el nostre text en particular. Per filtrar-les en el nostre cas son les `ent.label_ == 'FECHA'`

Amb aquest codi podem veure les dates trobades en el text:

```
for ent in document.ents:
      if ent.label_ == 'FECHA':
            print(ent.text)
```

10.5.4 Resultats

Les següents taules mostren els resultats dels models provats amb els diversos documents del *validation set.*

RESULTATS MODEL PETIT			
TEXTOS	METRIQUES		
	Precision	*Recall*	*F1-Score*
ej_0_es	0.95	0.95	0.95
ej_1_es	0.97	0.38	0.55
ej_2_es	1.00	0.94	0.97
ej_3_es	0.90	0.81	0.85
Mitjana dels textos	**0.96**	**0.77**	**0.83**

Taula 8 Resultats del rendiment del model petit

RESULTATS MODEL MITJÀ			
TEXTOS	METRIQUES		
	Precision	*Recall*	*F1-Score*
ej_0_es	0.97	0.91	0.94
ej_1_es	0.94	0.38	0.54
ej_2_es	1.00	0.94	0.97
ej_3_es	0.87	0.81	0.84
Mitjana dels textos	**0.95**	**0.76**	**0.82**

Taula 9 Resultats del rendiment del model mitjà

11. Conclusions

11.1 Expressions Regulars

El sistema és capaç de capturar totes les dates, sempre que el format d'aquestes hagin estat prèviament estudiants i afegit al conjunt de patrons que busquem en els textos.

Aquestes expressions regulars són molt complexes i es necessita un gran coneixement per tal de mantenir-les, actualitzar-les i millorar-les, provocant un greu problema en el cas que es vulguin mantenir, actualitzar o modificar.

11.2 spaCy

El sistema es capaç de capturar totes les dates, sempre que el format d'aquestes hagin estat prèviament estudiants i afegit al conjunt de patrons que busquem en els textos.

Els patrons que s'utilitzen en aquest sistema, no són tant complexos com els utilitzats en les expressions regulars, per la qual cosa, són fàcils d'optimitzar, mantenir, modificar i actualitzar. A més a més, en aquests patrons també es poden afegir expressions regulars, per la qual cosa aquest sistema podem dir que és un sistema híbrid.

11.3 Models

Per comparar els resultats obtinguts dels diversos sistemes s'han fet testos amb diferents documents els quals no han estat utilitzats en el "training set". En total aquests 4 textos amb els quals comprovem els resultats tenen 515 entitats de dates de les 2258 del total que hem utilitzat en aquest projecte. Per tant, el "validation set" és un 22.80% de les dades totals.

Els resultats dels models són comparats amb una matriu de confusió explicada en l'apartat **3.6 Metodologies d'avaluació del rendiment de models** i es considera que un test ha estat passar correctament si té un rendiment superior al 70% d'encert.

Seguidament, es poden observar els resultats de *Precision, Recall i F1-Score* en el nostre conjunt de textos del *validation set*:

Com es pot observar, el *Recall* i la *F1-Score* en el text ej_1_es és molt baix, això es degut que en aquest text es troben moltes dades en format 14/03/2015, 14.03.2015 i 14-03-2015 i els models no les troben del tot bé. Per totes les altres mètriques els resultats són més que satisfactoris.

Aquesta és la taula per comparar els diferents models testejats:

	PRECISION	**RECALL**	**F1-SCORE**
Model en blanc	0.94	0.73	0.81
Model petit	**0.96**	**0.77**	**0.83**
Model mitjà	0.95	0.76	0.82

Taula 10 Taula de rendiments dels tres models entrenats

Com es pot observar el millor model ha estat el que hem tornat a entrenat el model petit, però com podem observar, la diferència és pràcticament insignificant. Tenint en compte el *F1-SCORE* com a mètrica d'avaluació principal, podem observar que tenim un 83% d'encert en les nostres prediccions. D'altra banda, els nostres sistemes basats en regles, són capaços de torbar totes les dates, i en cas que alguna no la trobi, és pot afegir una nova regla.

Per concloure, tenint en compte tots els factors que he anat trobant a l'hora d'estudiar, implementar i testejar els sistemes els quals he pogut comparar. Concloc que el millor sistemes per trobar dates en textos legals és el sistema basat en regles d'spaCy, a causa de:

La hibridació que té amb les expressions regulars, les quals poden servir per cobrir els casos del ej_1_es citat anteriorment, el seu nivell de sofisticació i precisió que pot arribar a tenir gràcies al POS i la seva facilitat d'implementar-lo. Finalment afegir que entendre els patrons resulta molt més senzill que els que s'utilitzen en les expressions regular, això pot ajudar a optimitzar-les, si escau.

La temporalització del projecte duta a terme no ha variat gaire de la planificada inicialment, ja que s'estimava que l'acabaria a finals de juny i ha sigut a principis de juliol.

12. Treball Futur

El que més m'ha agradat d'aquest projecte ha estat descobrir el món del Processament del Llenguatge Natural i a més a més, descobrir les diferents tècniques de NER que existeixen i estudiar-les.

Aquest treball que he realitzat obre un gran ventall de possibilitats en les quals es pot avançar ja que ara tenim uns resultats que ens poden ajudar a escollir quina és la millor manera de trobar entitats.

Hi ha una llibreria la qual és Blackstone[16] que serveis per processar text legal en anglès.

- És el primer model de codi obert capacitat per al seu ús en textos de gran extensió que contenen entitats i conceptes en el context del dret legal.
- Es basa en spaCy, per la qual cosa facilita la recol·lecció i aplicació de les seves pròpies dades.
- És gratuït i de codi obert.
- És imperfecte.

Com he dit, aquest projecte que he realitzat m'ha ajudat a entendre quina estratègia és millor per trobar entitats en textos legals, és per això que ara el següent pas seria fer una llibreria i un model propi per trobar noves entitats. Cosa que és el que estava realitzant amb el meu equip a Indra, ja que també trobàvem quantitats, terminis, el tipus de resolució...

[16] Pàgina web: https://research.iclr.co.uk/blackstone

13. Bibliografia

- Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, *12*(ARTICLE), 2493-2537.
- Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., & McClosky, D. (2014, June). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations* (pp. 55-60).
- Klein, E., & Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".
- Manning, C., & Schutze, H. (1999). *Foundations of statistical natural language processing*. MIT press.
- Indurkhya, N., & Damerau, F. J. (Eds.). (2010). *Handbook of natural language processing* (Vol. 2). CRC Press.
- Chiu, J. P., & Nichols, E. (2016). Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, *4*, 357-370.
- Florian, R., Ittycheriah, A., Jing, H., & Zhang, T. (2003). Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003* (pp. 168-171).
- Nadeau, D., & Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, *30*(1), 3-26.
- Dozier, C., Kondadadi, R., Light, M., Vachher, A., Veeramachaneni, S., & Wudali, R. (2010). Named entity recognition and resolution in legal text. In *Semantic Processing of Legal Texts* (pp. 27-43). Springer, Berlin, Heidelberg.
- de Araujo, P. H. L., de Campos, T. E., de Oliveira, R. R., Stauffer, M., Couto, S., & Bermejo, P. (2018, September). Lener-br: a dataset for named entity recognition in brazilian legal text. In *International Conference on Computational Processing of the Portuguese Language* (pp. 313-323). Springer, Cham.
- Kosinov, S., Kozintsev, I., Polito, M., & Dulong, C. (2008). *U.S. Patent Application No. 11/508,579*.
- Dozier, C., Kondadadi, R., Light, M., Vachher, A., Veeramachaneni, S., & Wudali, R.(2010). Named entity recognition and resolution in legal text. In Semantic Processingof Legal Texts (pp. 27-43). Springer, Berlin, Heidelberg.
- Nikoulina, V., & Sandor, A. (2014). *U.S. Patent Application No. 13/707,745*.
- Partalidou, E., Spyromitros-Xioufis, E., Doropoulos, S., Vologiannidis, S., & Diamantaras, K. I. (2019, October). Design and implementation of an open source Greek POS Tagger and Entity Recognizer using spaCy. In *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)* (pp. 337-341). IEEE.
- Dang, T. H., Le, H. Q., Nguyen, T. M., & Vu, S. T. (2018). D3NER: biomedical named entity recognition using CRF-biLSTM improved with fine-tuned embeddings of various linguistic information. *Bioinformatics*, *34*(20), 3539-3546.
- Konkol, M. (2012). Named entity recognition: technical report no. DCSE/TR-2012-04.

- Surdeanu, M., Nallapati, R., & Manning, C. (2010, May). Legal claim identification:Information extraction with hierarchically labeled data. In Workshop Programme (p.22).

14. Annexes

14.1 Matrius de confusió dels models

Seguidament es mostren en format de taula les matrius de confusió dels resultats que s'expliquen en l'apartat: 11. Conclusions

Com seguidament es pot observar el valor **True Negative (TN)** no estan informats en les taules, això és degut al fet que aquests correspondrien a totes les entitats que potencialment es podrien identificar com dates. A més a més, les nostres mètriques de mesura no ho utilitzen.

Matriu de confusió		Valors actuals	
		Positiu(1)	**Negatiu(0)**
Valors predits	**Positiu(1)**	TP	FP
	Negatiu(0)	FN	TN

Taula 11 Estructura de la matriu de confusió

Matriu de confusió		Valors actuals	
		Positiu(1)	**Negatiu(0)**
Valors predits	**Positiu(1)**	37	0
	Negatiu(0)	6	-

Taula 12 Matriu de confusió del text ej_0_es i el model en blanc

Matriu de confusió		Valors actuals	
		Positiu(1)	**Negatiu(0)**
Valors predits	**Positiu(1)**	30	3
	Negatiu(0)	49	-

Taula 13 Matriu de confusió del text ej_1_es i el model en blanc

Matriu de confusió		Valors actuals	
		Positiu(1)	**Negatiu(0)**
Valors predits	**Positiu(1)**	33	0
	Negatiu(0)	2	-

Taula 14 Matriu de confusió del text ej_2_es i el model en blanc

Matriu de confusió		Valors actuals	
		Positiu(1)	**Negatiu(0)**
Valors predits	**Positiu(1)**	264	46
	Negatiu(0)	93	-

Taula 15 Matriu de confusió del text ej_3_es i el model en blanc

Matriu de confusió		Valors actuals	
		Positiu(1)	**Negatiu(0)**
Valors predits	**Positiu(1)**	41	2
	Negatiu(0)	2	-

Taula 16 Matriu de confusió del text ej_0_es i el model petit

Matriu de confusió		Valors actuals	
		Positiu(1)	**Negatiu(0)**
Valors predits	**Positiu(1)**	30	1
	Negatiu(0)	49	-

Taula 17 Matriu de confusió del text ej_1_es i el model petit

Matriu de confusió		Valors actuals	
		Positiu(1)	**Negatiu(0)**
Valors predits	**Positiu(1)**	33	0
	Negatiu(0)	2	-

Taula 18 Matriu de confusió del text ej_2_es i el model petit

Matriu de confusió		Valors actuals	
		Positiu(1)	**Negatiu(0)**
Valors predits	**Positiu(1)**	288	31
	Negatiu(0)	69	-

Taula 19 Matriu de confusió del text ej_3_es i el model petit

Matriu de confusió		Valors actuals	
		Positiu(1)	**Negatiu(0)**
Valors predits	**Positiu(1)**	39	1
	Negatiu(0)	4	-

Taula 20 Matriu de confusió del text ej_0_es i el model mitjà

Matriu de confusió		Valors actuals	
		Positiu(1)	**Negatiu(0)**
Valors predits	**Positiu(1)**	30	2
	Negatiu(0)	49	-

Taula 21 Matriu de confusió del text ej_1_es i el model mitjà

Matriu de confusió		Valors actuals	
		Positiu(1)	**Negatiu(0)**
Valors predits	**Positiu(1)**	33	0
	Negatiu(0)	2	-

Taula 22 Matriu de confusió del text ej_2_es i el model mitjà

Matriu de confusió		Valors actuals	
		Positiu(1)	**Negatiu(0)**
Valors predits	**Positiu(1)**	290	44
	Negatiu(0)	67	-

Taula 23 Matriu de confusió del text ej_3_es i el model mitjà

14.2 Mètriques d'entrenament dels models

Els models han estat entrenats amb un ordenador de sobretaula amb les següent característiques:

Processador: Intel Core i7-6700 a 3.40GHz

Memòria RAM: 16 GB

S.O: Windows 10 PRO

14.2.1 Model petit (es_core_news_sm)

Aquest models té les següents característiques:

es_core_news_sm	
LLENGUATGE	Espanyol
TIPUS	Vocabulari, sintaxi, entitats i vectors
GÈNERE	Text escrit (notícies, i mitjans de comunicació)
TAMANY	15 MB
PIPELINE	Tagger, parser, ner
VECTORS	n/a
RECURSOS	UD Spanish AnCora v2.5 (Martínez Alonso, Héctor; Zeman, Daniel) WikiNER
AUTOR	Explosion
LLICÈNCIA	GPL

Taula 24 Característiques model es_core_news_sm

Precisió de sintaxi	
Dependències etiquetades	87.43
Dependències no etiquetades	90.76
Part-of-speech tags	97.18

Taula 25 Precisió de la sintaxi del model es_core_news_sm

Precisió de NER	
NER F1-SCORE	89.41
NER PRECISION	89.55
NER RECALL	89.27

Taula 26 Precisió de NER del model es_core_news_sm

Temps de reentrenament del model petit amb la nova entitat: 3 hores i 13 minuts.

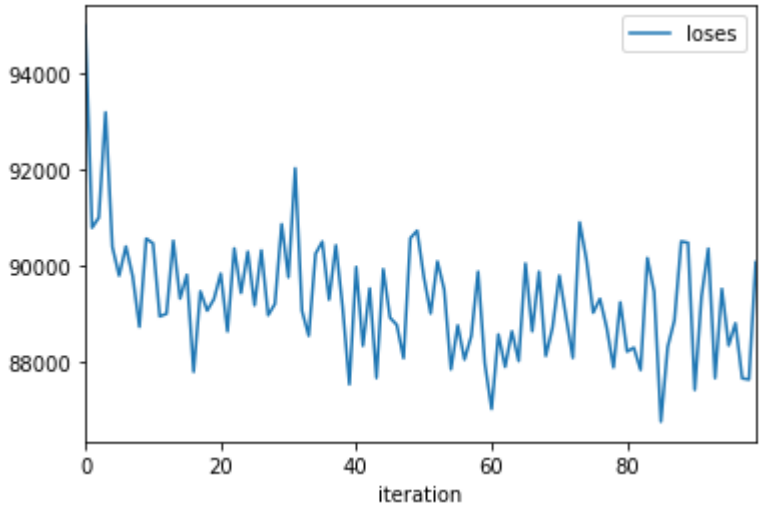

Figura 14 Relació Loses/Iteracions entrenament del model es_core_legal_sm

14.2.2 Model mitjà (es_core_news_md)

es_core_news_sm	
LLENGUATGE	Espanyol
TIPUS	Vocabulari, sintaxi, entitats i vectors
GÈNERE	Text escrit (notícies, i mitjans de comunicació)
TAMANY	45 MB
PIPELINE	Tagger, parser, ner
VECTORS	500k claus, 20k vectors únics (300 dimensions)
RECURSOS	UD Spanish AnCora v2.5 (Martínez Alonso, Héctor; Zeman, Daniel) WikiNER OSCAR Wikipedia(20200301)
AUTOR	Explosion
LLICÈNCIA	GPL

Taula 27 Característiques model es_core_news_md

Precisió de sintaxi	
Dependències etiquetades	87.43
Dependències no etiquetades	90.76
Part-of-speech tags	97.18

Taula 28 Precisió de la sintaxi del model es_core_news_md

Precisió de NER	
NER F1-SCORE	89.41
NER PRECISION	89.55
NER RECALL	89.27

Taula 29 Precisió de NER del model es_core_news_sm

Temps de reentrenament del model mitjà amb la nova entitat: 3 hores i 8 minuts.

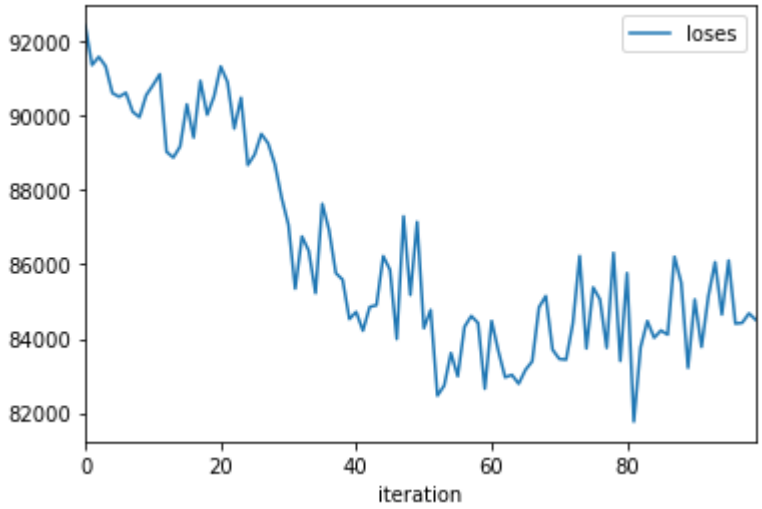

Figura 15 Relació Loses/Iteracions entrenament del model es_core_legal_md

14.2.3 Model en blanc

Temps d'entrenament: 4 hores i 12 minuts.

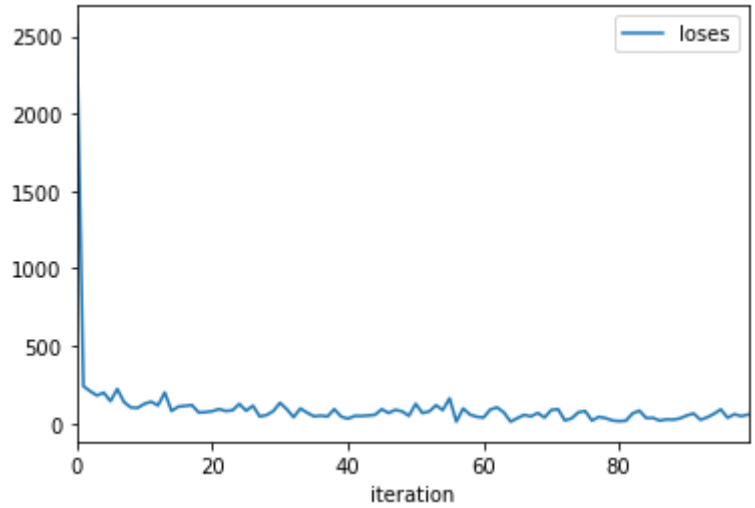

Figura 16 Relació Loses/Iteracions entrenament del model en blanc